Package: pacheck (via r-universe)

November 6, 2026

Type Package

Title Probabilistic Analysis Check Package

Version 0.2.2.9000

Maintainer Xavier Pouwels <x.g.l.v.pouwels@utwente.nl>

Author Xavier Pouwels [aut, cre, cph]

Description Investigate (analytically or visually) the inputs and outputs of probabilistic analyses of health economic models using standard health economic visualisation and metamodelling methods.

License GPL (>= 3)

Imports assertthat, boot, dplyr, fitdistrplus, flexsurv, ggplot2, glue, gtools, interp, moments, randomForestSRC, reshape2, scales, signal, simsurv, stats, stringi, testthat, tidyr, glmnet, tibble, survival

Config/testthat/edition 3

Encoding UTF-8

LazyDataCompression xz

LazyData true

RoxygenNote 7.3.2

URL https://xa4p.github.io/pacheck/, https://github.com/Xa4P/pacheck

Suggests knitr, rmarkdown

Depends R (>= 4.1.0)

VignetteBuilder knitr

BugReports https://github.com/Xa4P/pacheck/issues

Repository https://xa4p.r-universe.dev

Date/Publication 2026-02-09 11:56:26 UTC

RemoteUrl https://github.com/xa4p/pacheck

RemoteRef HEAD

RemoteSha 2dd584c5a2e98cd8cb12218d95a67d960f9816c8

Contents

calculate_ceac . . . 3
calculate_ceac_mult . . . 4
calculate_nb . . . 5
calculate_nb_mult . . . 6
check_binary . . . 7
check_mean_qol . . . 8
check_positive . . . 8
check_psa_darth . . . 9
check_range . . . 10
check_sum_probs . . . 11
check_sum_vars . . . 12
check_surv_mod . . . 13
df_ckd_inputs . . . 14
df_ckd_results . . . 14
df_iviRA_pa . . . 15
df_pa . . . 16
df_pa_psm . . . 18
do_check . . . 20
do_discount_check . . . 21
do_quick_check . . . 21
estimate_decision_sensitivity . . . 22
fit_dist . . . 23
fit_lasso_metamodel . . . 24
fit_lm_metamodel . . . 26
fit_rf_metamodel . . . 28
generate_cor . . . 30
generate_det_inputs . . . 31
generate_pa_inputs . . . 31
generate_pa_inputs_psm . . . 32
generate_sum_stats . . . 33
l_psa_aaa . . . 34
perform_dowsa . . . 34
perform_simulation . . . 35
perform_simulation_psm . . . 36
plot_ce . . . 36
plot_ce_mult . . . 37
plot_ceac . . . 38
plot_convergence . . . 39
plot_ice . . . 40
plot_nb . . . 41
plot_nb_mult . . . 42
plot_surv_mod . . . 43
predict_metamodel . . . 44
summary_ice . . . 45
validate_metamodel . . . 46
vis_1_param . . . 47

vis_2_params 48

Index **50**

`calculate_ceac` *Calculate cost-effectiveness probabilities for two strategies.*

Description

This function calculates the probabilities that each strategy is the cost effective at different willingness to pay thresholds.

Usage

```
calculate_ceac(
  df,
  e_int,
  e_comp,
  c_int,
  c_comp,
  v_wtp = seq(from = 0, to = 1e+05, by = 1000)
)
```

Arguments

`df`	a dataframe.
`e_int`	character. Name of variable of the dataframe containing total effects of the intervention strategy.
`e_comp`	character. Name of variable of the dataframe containing total effects of the comparator strategy.
`c_int`	character. Name of variable of the dataframe containing total costs of the intervention strategy.
`c_comp`	character. Name of variable of the dataframe containing total costs of the comparator strategy.
`v_wtp`	vector of numerical values. Vector of willingness-to-pay threshold for which the probabilities of cost effectiveness have to be defined. Default is 0:100,000 by increments of 1,000.

Value

A dataframe with three columns:

- WTP_threshold = The willingness-to-pay thresholds at which the probability of cost effectiveness has been calculated for both strategies
- Prob_int = The probability that the intervention strategy is cost effective at a given willingness-to-pay threshold
- Prob_comp = The probability that the comparator strategy is cost effective at a given willingness-to-pay threshold

Examples

```
# Calculate probabilities of cost effectiveness using the example dataframe,
# for willlingness-to-pay thresholds of 0 to 50,0000 euros.
data("df_pa")
calculate_ceac(df = df_pa,
               e_int = "t_qaly_d_int",
               e_comp = "t_qaly_d_comp",
               c_int = "t_costs_d_int",
               c_comp = "t_costs_d_comp",
               v_wtp = seq(from = 0, to = 50000, by = 1000))
```

`calculate_ceac_mult` *Calculate cost-effectiveness probabilities.*

Description

This function calculates the probabilities that each strategy is the cost effective at different willingness to pay thresholds, for an infinite amount of strategies.

Usage

```
calculate_ceac_mult(
  df,
  outcomes,
  costs,
  v_wtp = seq(from = 0, to = 1e+05, by = 1000)
)
```

Arguments

`df`	a dataframe.
`outcomes`	character. Vector of variable names containing the outcomes to be plotted on the x-axis. The variable names should be structured as follows: 't_qaly_d_' followed by the name of the strategy: e.g. 't_qaly_d_intervention'.
`costs`	character. Vector of variable names containing the costs to be plotted on the y-axis. The variable names should be structured as follows: 't_costs_d_' followed by the name of the strategy: e.g. 't_costs_d_intervention'.
`v_wtp`	vector of numerical values. Vector of willingness-to-pay threshold for which the probabilities of cost effectiveness have to be defined. Default is 0:100,000 by increments of 1,000.

Value

A dataframe with three columns:

- WTP_threshold = The willingness-to-pay thresholds at which the probability of cost effectiveness has been calculated for both strategies

- Prob_int = The probability that the intervention strategy is cost effective at a given willingness-to-pay threshold
- Prob_comp = The probability that the comparator strategy is cost effective at a given willingness-to-pay threshold

Examples

```
# Calculate probabilities of cost effectiveness using the example dataframe,
data("df_pa")
df_pa$t_qaly_d_int2 <- df_pa$t_qaly_d_int * 1.5 # creating additional outcome variable
df_pa$t_costs_d_int2 <- df_pa$t_costs_d_int * 1.5 # creating additional cost variable
calculate_ceac_mult(df = df_pa,
             outcomes = c("t_qaly_d_int", "t_qaly_d_comp", "t_qaly_d_int2"),
             costs = c("t_costs_d_int","t_costs_d_comp", "t_costs_d_int2")
             )
```

---

calculate_nb *Calculate NMB and NHB for two strategies.*

Description

This function calculates the Net Monetary Benefits (NMB) and Net Health Benefits (NHB) for each strategy and the incremental NMB and NHB.

Usage

```
calculate_nb(df, e_int, e_comp, c_int, c_comp, wtp)
```

Arguments

df	a dataframe.
e_int	character. Name of variable of the dataframe containing total effects of the intervention strategy.
e_comp	character. Name of variable of the dataframe containing total effects of the comparator strategy.
c_int	character. Name of variable of the dataframe containing total costs of the intervention strategy.
c_comp	character. Name of variable of the dataframe containing total costs of the comparator strategy.
wtp	numeric. Willingness-to-pay thresholds to use for NMB and NHB calculations.

Value

A dataframe containing the original data and the following variables:

- NMB_int = Net monetary benefit of the intervention
- NMB_comp = Net monetary benefit of the comparator
- iNMB = Incremental net monetary benefit of the intervention versus the comparator
- NHB_int = Net health benefit of the intervention
- NHB_comp = Net health benefit of the comparator
- iNHB = Net health benefit of the intervention versus the comparator

Examples

```
# Calculate NB's at a willingness-to-pay threshold of 80000 per unit of effects
data("df_pa")
calculate_nb(df = df_pa,
             e_int = "t_qaly_d_int",
             e_comp = "t_qaly_d_comp",
             c_int = "t_costs_d_int",
             c_comp = "t_costs_d_comp",
             wtp = 80000)
```

---

calculate_nb_mult *Calculate NMB and NHB.*

Description

This function calculates the Net Monetary Benefits (NMB) and Net Health Benefits (NHB) for each strategy and the incremental NMB and NHB.

Usage

```
calculate_nb_mult(df, outcomes, costs, wtp)
```

Arguments

df	a dataframe.
outcomes	character. Vector of variable names containing the outcomes to be plotted on the x-axis. The variable names should be structured as follows: 't_qaly_d_' followed by the name of the strategy: e.g. 't_qaly_d_intervention'.
costs	character. Vector of variable names containing the costs to be plotted on the y-axis. The variable names should be structured as follows: 't_costs_d_' followed by the name of the strategy: e.g. 't_costs_d_intervention'.
wtp	numeric. Willingness-to-pay thresholds to use for NMB and NHB calculations.

Value

A dataframe containing the original data and the following variables containing the NMB and NHB for each strategy. The name of these new variables are structured as 'NMB_strategyname' and 'NHB_strategyname'

Examples

```
# Calculate NB's at a willingness-to-pay threshold of 80000 per unit of effects
data("df_pa")
df_pa$t_costs_d_comp2 <- df_pa$t_costs_d_comp * 1.09
df_pa$t_qaly_d_comp2 <- df_pa$t_qaly_d_comp * 1.01
calculate_nb_mult(df = df_pa,
                  outcomes = c("t_qaly_d_comp2", "t_qaly_d_int", "t_qaly_d_comp"),
                  costs = c("t_costs_d_int", "t_costs_d_comp2", "t_costs_d_comp"),
                  wtp = 50000
                  )
```

`check_binary` *Check binary*

Description

This function tests whether the value of variables remain between 0 and 1 (for instance for utility and probability inputs)

Usage

```
check_binary(..., df, max_view = 50)
```

Arguments

`...`	character vector. This character vector contains the name of the variables of which the sum will be checked.
`df`	a dataframe.
`max_view`	numeric. Determines the number of iterations to display which do not fulfill the check. Default is 50.

Value

A dataframe.

Examples

```
# Checking whether a variable is strictly positive
data(df_pa)
check_binary("u_pfs", df = df_pa)
# Checking whether two variables are strictly positive
# Decreasing the number of iterations to display to 20.
check_binary("u_pfs", "p_pfspd", df = df_pa)
```

check_mean_qol *Check mean quality of life*

Description

This function checks whether the mean quality of life outcome of each iteration remain between the maximum and minimum utility values of the specific iteration.

Usage

```
check_mean_qol(df, t_qaly, t_ly, u_values, max_view = 100)
```

Arguments

df	a dataframe.
t_qaly	character. Name of the variable containing the total undiscounted quality-adjusted life years.
t_ly	character. Name of the variable containing the total undiscounted life years.
u_values	(vector of) character. Name(s) of the variable containing the utility values.
max_view	numeric. Determines the number of iterations to display which do not fulfil the check. Default is 100.

Value

A matrix.

Examples

```
# Check whether mean quality of life is within min-max utility values
check_mean_qol(df = df_pa,
               t_ly = "t_ly_comp",
               t_qaly = "t_qaly_comp",
               u_values = c("u_pfs", "u_pd")
               )
```

check_positive *Check whether variables are strictly positive*

Description

This function tests whether variables are strictly positive (for instance for costs and relative risks inputs)

Usage

```
check_positive(..., df, max_view = 50)
```

Arguments

...	character vector. This character vector contains the name of the variables of which the sum will be checked.
df	a dataframe.
max_view	numeric. Determines the number of iterations to display which do not fulfill the check. Default is 50.

Value

A dataframe.

Examples

```
# Checking whether a variable is strictly positive
check_positive("c_pfs", df = df_pa)

# Checking whether two variables are strictly positive
# Descreasing the number of iterations to display to 20.
check_positive("c_pfs", "c_pd", df = df_pa)
```

---

check_psa_darth *Check PSA inputs & outputs*

Description

This function checks whether the value of variables remain between 0 and 1 for utility and probability inputs, and are strictly positive for costs, hazard ratios, odds ratios, relative risks, and total outcomes of each strategy.

Usage

```
check_psa_darth(
  l_psa_darth,
  utility = "u_",
  costs = "c_",
  probs = "p_",
  rr = "rr_",
  hr = "hr_",
  or = "or_",
  exclude = NULL,
  v_outcome = c("effectiveness", "cost")
)
```

Arguments

l_psa_darth	a list of class 'psa' as obtained by the function [dampack::make_psa_obj()]
utility	characters. String used at the start of the variables identifying utility inputs.
costs	characters. String used at the start of the variables identifying cost inputs.
probs	characters. String used at the start of the variables identifying probability inputs.
rr	characters. String used at the start of the variables identifying relative risk inputs.
hr	characters. String used at the start of the variables identifying hazard ratio inputs.
or	characters. String used at the start of the variables identifying odds ratio inputs.
exclude	vector of strings. Vector containing the name of the input variables not to include in the checks. Default is NULL, hence all variables from the 'parameters' dataframe are included.
v_outcome	vector of strings. Vector containing the name of the output variables to include in the checks. Default values are 'effectiveness' and 'cost'.

Value

A matrix containing the input and output variables that have been checked and the iterations wherein an erroneous value has been identified.

---

check_range *Check range*

Description

This function tests whether an input or output value falls within a user-defined range and return the proportion of iteration in which this is not the case.

Usage

```
check_range(df, param, min_val = NULL, max_val = NULL)
```

Arguments

df	a dataframe.
param	character string. Name of variable of the dataframe for which to check the range.
min_val	numeric. Define the minimum value of the range.
max_val	numeric. Define the maximum value of the range.

Details

If only 'min_val' is specified, the proportion of iteration above this value will be computed. If only 'max_val' is specified, the proportion of iteration below this value will be computed.

Value

A numeric.

Examples

```
# Checking how often the "u_pfs" values falls within 0.55 and 0.72.
data(df_pa)
check_range(df = df_pa,
            param = "u_pfs",
            min_val = 0.55,
            max_val = 0.72
                 )
```

`check_sum_probs` *Check sum probabilities*

Description

This function checks whether the sum of user-defined variables representing probabilities is below or equal to 1 for each iteration of the probabilistic inputs.

Usage

```
check_sum_probs(..., df, digits = NULL, check = "lower", max_view = 100)
```

Arguments

`...`	character vector. This character vector contains the name of the variables of which the sum will be checked.
`df`	a dataframe.
`digits`	numeric. Define the number of digits at which the sum of probabilities should be rounded.
`check`	logical. Define which test to perform."lower" tests whether the sum of the selected variables is lower than or equal to 1 for each iteration. "equal" tests whether the sum of the selected variables is equal to 1 for each iteration. Default is "lower".
`max_view`	numeric. Determines the number of iterations to display which do not fulfill the test Default is 100.

Value

A text indicating whether the sum of the probabilities is belor and/or eual to one or indicating in which iteration that is not the case.

Examples

```
# Checking whether the sum of the two probabilities is lower than or equal to 1
check_sum_probs("p_pfspd", "p_pfsd", df = df_pa, check = "lower")

# Checking the sum of the two probabilities equals 1 using a vector to select them,
# Rounding off to two digits, and extending the number of iterations to display to 250.
check_sum_probs(c("p_pfspd", "p_pfsd"), df = df_pa, digits = 2, check = "equal", max_view = 250)
```

`check_sum_vars` *Check sum variables*

Description

This function tests whether the sum of selected variables are equal to another.

Usage

```
check_sum_vars(..., df, outcome, digits = 3)
```

Arguments

`...`	character vector. This character vector contains the name of the variables of which the sum will be checked.
`df`	a dataframe.
`outcome`	character string. Name of variable of the dataframe which should equal the sum of variables mentioned in '...'.
`digits`	Define the number of digits at which the sum and the 'outcome' variables are rounded. Default is 3 digits.

Value

A string.

Examples

```
# Checking whether health state and adverse event costs equal the total discounted costs
check_sum_vars("t_costs_pfs_d_int", "t_costs_pd_d_int", "t_costs_ae_int",
               df = head(df_pa),
               outcome = "t_costs_d_int",
               digits = 0)
```

check_surv_mod *Check parametric survival models*

Description

This function tests whether the first of two parametric survival model is lower than a second parametric survival model.

Usage

```
check_surv_mod(
  df,
  surv_mod_1,
  surv_mod_2,
  v_names_param_mod_1,
  v_names_param_mod_2,
  time = seq(0, 5, 0.1),
  label_surv_1 = "first survival",
  label_surv_2 = "second survival",
  n_view = 10
)
```

Arguments

df	a dataframe.
surv_mod_1	character. Name of the parametric model to use for the first survival model.
surv_mod_2	character. Name of the parametric model to use for the second survival model.
v_names_param_mod_1	(vector of) character. Name of the columns containing the parameter values for the first survival model.
v_names_param_mod_2	(vector of) character. Name of the columns containing the parameter values for the second survival model.
time	a numerical vector. Determine at which time points survival probabilities have to be estimated for both survival models. For each of these time points, it will be checked whether the first survival model results in higher survival probabilities than the second survival model.
label_surv_1	character vector. The label to provide to the first survival curve (relevant for export).
label_surv_2	character vector. The label to provide to the second survival curve (relevant for export).
n_view	integer. Number of iterations to mention in which the curves are crossing. Default is 10.

Details

The parametric models that can be used are the following: exponential (exp), Weibull (weibull), gamma (gamma), loglogistic (logis), and lognormal (lnorm). All these functions are implemented following their distribution function as documented in the stats package.

Value

A list. The first element is a message, the second element contains the number of the iterations in which the the first curve is higher than the second curve.

---

df_ckd_inputs *A dataframe containing probabilistic inputs for testing*

Description

A dataframe containing 1,000 sets of probabilistic inputs from a health state transition model developed in Python.

Usage

```
df_ckd_inputs
```

Format

The dataframe contains the 1,000 probabilistic inputs (indexes 1 to 1000) from the PSA.xlsx file available on the github repository of the model.

Source

Marika M. Cusick, Rebecca L. Tisdale, Glenn M. Chertow, et al. Population-Wide Screening for Chronic Kidney Disease: A Cost-Effectiveness Analysis. Ann Intern Med.2023;176:788-797. [Epub 23 May 2023]. doi:10.7326/M22-3228.

Link to Github repository: https://github.com/marikamaecusick/CKDScreeningCEA

---

df_ckd_results *A dataframe containing probabilistic outputs for testing*

Description

A dataframe containing 1,000 sets of probabilistic outputs from a health state transition model developed in Python for individuals aged 35, screened every 10 years, versus no screening.

Usage

```
df_ckd_results
```

Format

The dataframe contains the 1,000 probabilistic outputs from the CKD Screening CEA health economic model.

Source

Marika M. Cusick, Rebecca L. Tisdale, Glenn M. Chertow, et al. Population-Wide Screening for Chronic Kidney Disease: A Cost-Effectiveness Analysis. Ann Intern Med.2023;176:788-797. [Epub 23 May 2023]. doi:10.7326/M22-3228.

Link to Github repository: https://github.com/marikamaecusick/CKDScreeningCEA

---

df_iviRA_pa *Dataframe of inputs and outputs of a health economic model developed and evaluated with the iviRA R package for testing*

Description

A dataframe containing 1,000 sets of probabilistic inputs and outputs of a individual-level health state transition model developed using iviRA package available on Github.

Usage

```
df_iviRA_pa
```

Format

See the documentation of the ‘sample_pars‘ function of the iviRA package for details.

Source

Incerti D, Curtis JR, Shafrin J, Lakdawalla DN, Jansen JP (2019). “ A Flexible Open-Source Decision Model for Value Assessment of Biologic Treatment for Rheumatoid Arthritis.” PharmacoEconomics. doi: 10.1007/s40273-018-00765-2.

Link to Github repository: https://innovationvalueinitiative.github.io/IVI-RA/index.html

Link to iviRA website: https://innovationvalueinitiative.github.io/IVI-RA/

df_pa *Dataframe for testing*

Description

A dataframe containing 1,000 iterations of a probabilistic analysis of a health state transition model. To access the original dataframe used in the scientific publication of PACBOARD (Pouwels et al. 2024), follow the link below.

Usage

```
df_pa
```

Format

A dataframe with 1,000 rows, each row being the inputs and (intermediate) outputs of a single probabilistic iteration, and 44 variables:

p_pfspd Probability to transit from the progression-free survival (PFS) to progressed disease (PD) health state

p_pfsd Probability to transit from the PFS to Death (D) health state

p_pdd Probability to transit from the PD to D health state

p_dd Probability to transit from the D to D health state

p_ae Probability of occurence of an adverse event in the intervention stategy

rr Relative effectiveness of the treatment (_int)

u_pfs Utility value (per cycle) associated with PFS health state

u_pd Utility value (per cycle) associated with PD health state

u_d Utility value (per cycle) associated with D health state

u_ae Utility decrement associated with the occurrence of an adverse event

c_pfs Costs (per cycle) associated with PFS health state

c_pd Costs (per cycle) associated with PD health state

c_d Costs (per cycle) associated with D health state

c_thx Costs (per cycle) associated with receiving treatment, in the PFS health state

c_ae Costs associated with experiencing an adverse event

t_qaly_comp Total undiscounted QALY obtained with the comparator, i.e. no treatment administered

t_qaly_int Total undiscounted QALY obtained with the intervention, i.e. treatment administered

t_qaly_d_comp Total discounted QALY obtained with the comparator, i.e. no treatment administered

t_qaly_d_int Total discounted QALY obtained with the intervention, i.e. treatment administered

t_costs_comp Total undiscounted costs obtained with the comparator, i.e. no treatment administered

t_costs_int Total undiscounted costs obtained with the intervention, i.e. treatment administered

t_costs_d_comp Total discounted costs obtained with the comparator, i.e. no treatment administered

t_costs_d_int Total discounted costs obtained with the intervention, i.e. treatment administered

t_ly_comp Total undiscounted LY obtained with the comparator, i.e. no treatment administered

t_ly_int Total undiscounted LY obtained with the intervention, i.e. treatment administered

t_ly_d_comp Total discounted LY obtained with the comparator, i.e. no treatment administered

t_ly_d_int Total discounted LY obtained with the intervention, i.e. treatment administered

t_ly_pfs_d_comp Total discounted life years accrued in PFS health state, comparator strategy

t_ly_pfs_d_int Total discounted life years accrued in PFS health state, intervention strategy

t_ly_pd_d_comp Total discounted life years accrued in PD health state, comparator strategy

t_ly_pd_d_int Total discounted life years accrued in PD health state, intervention strategy

t_qaly_pfs_d_comp Total discounted quality-adjusted life years accrued in PFS health state, comparator strategy

t_qaly_pfs_d_int Total discounted quality-adjusted life years accrued in PFS health state, intervention strategy

t_qaly_pd_d_comp Total discounted quality-adjusted life years accrued in PD health state, comparator strategy

t_qaly_pd_d_int Total discounted quality-adjusted life years accrued in PD health state, intervention strategy

t_costs_pfs_d_comp Total discounted costs accrued in PFS health state, comparator strategy

t_costs_pfs_d_int Total discounted costs accrued in PFS health state, intervention strategy

t_costs_pd_d_comp Total discounted costs accrued in PD health state, comparator strategy

t_costs_pd_d_int Total discounted costs accrued in PD health state, intervention strategy

t_qaly_ae_int Quality-adjusted life year decrement associated with the occurence of adverse events, intervention strategy

t_costs_ae_int Costs associated with the occurence of adverse events, intervention strategy

inc_ly Incremental QALYs obtained with the intervention versus the comparator

inc_qaly Incremental QALYs obtained with the intervention versus the comparator

inc_costs Incremental costs obtained with the intervention versus the comparator

Source

Pouwels XGLV, Kroeze K, van der Linden N, Kip MMA, Koffijberg H. Validating Health Economic Models With the Probabilistic Analysis Check dashBOARD. Value Health. 2024 Aug;27(8):1073-1084. doi: 10.1016/j.jval.2024.04.008.

Link to the original data ("df_pa") used in the PACBOARD publication: https://github.com/Xa4P/pacheck/tree/master/data-raw

`df_pa_psm` *Dataframe for testing*

Description

A dataframe containing 10,000 iterations of a probabilistic analysis of a partitioned survival model. To access the original dataframe used in the scientific publication of PACBOARD (Pouwels et al. 2024), follow the link below.

Usage

```
df_pa_psm
```

Format

A dataframe with 10,000 rows, each row being the inputs and (intermediate) outputs of a single probabilistic iteration, and 46 variables:

p_ae Probability of occurence of an adverse event in the intervention stategy

r_exp_pfs_comp Rate of the exponential survival model used to estimate PFS of the comparator

rr_thx_pfs Relative risk of the occurrence of progression of the intervention versus the comparator, used to estimate PFS of the intervention

r_exp_pfs_int Rate of the exponential survival model used to estimate PFS of the intervention

shape_weib_os Shape of the Weibull survival model used to estimate OS of the comparator and intervention

scale_weib_os_comp Scale of the Weibull survival model used to estimate OS of the comparator

rr_thx_os Relative risk of the occurrence of death of the intervention versus the comparator, used to estimate PFS of the intervention

scale_weib_os_int Scale of the Weibull survival model used to estimate OS of the intervention

u_pfs Utility value (per cycle) associated with PFS health state

u_pd Utility value (per cycle) associated with PD health state

u_d Utility value (per cycle) associated with D health state

u_ae Utility decrement associated with the occurrence of an adverse event

c_pfs Costs (per cycle) associated with PFS health state

c_pd Costs (per cycle) associated with PD health state

c_d Costs (per cycle) associated with D health state

c_thx Costs (per cycle) associated with receiving treatment, in the PFS health state

c_ae Costs associated with experiencing an adverse event

t_qaly_comp Total undiscounted QALY obtained with the comparator, i.e. no treatment administered

t_qaly_int Total undiscounted QALY obtained with the intervention, i.e. treatment administered

t_qaly_d_comp Total discounted QALY obtained with the comparator, i.e. no treatment administered

t_qaly_d_int Total discounted QALY obtained with the intervention, i.e. treatment administered

t_costs_comp Total undiscounted costs obtained with the comparator, i.e. no treatment administered

t_costs_int Total undiscounted costs obtained with the intervention, i.e. treatment administered

t_costs_d_comp Total discounted costs obtained with the comparator, i.e. no treatment administered

t_costs_d_int Total discounted costs obtained with the intervention, i.e. treatment administered

t_ly_comp Total undiscounted LY obtained with the comparator, i.e. no treatment administered

t_ly_int Total undiscounted LY obtained with the intervention, i.e. treatment administered

t_ly_d_comp Total discounted LY obtained with the comparator, i.e. no treatment administered

t_ly_d_int Total discounted LY obtained with the intervention, i.e. treatment administered

t_ly_pfs_d_comp Total discounted life years accrued in PFS health state, comparator strategy

t_ly_pfs_d_int Total discounted life years accrued in PFS health state, intervention strategy

t_ly_pd_d_comp Total discounted life years accrued in PD health state, comparator strategy

t_ly_pd_d_int Total discounted life years accrued in PD health state, intervention strategy

t_qaly_pfs_d_comp Total discounted quality-adjusted life years accrued in PFS health state, comparator strategy

t_qaly_pfs_d_int Total discounted quality-adjusted life years accrued in PFS health state, intervention strategy

t_qaly_pd_d_comp Total discounted quality-adjusted life years accrued in PD health state, comparator strategy

t_qaly_pd_d_int Total discounted quality-adjusted life years accrued in PD health state, intervention strategy

t_costs_pfs_d_comp Total discounted costs accrued in PFS health state, comparator strategy

t_costs_pfs_d_int Total discounted costs accrued in PFS health state, intervention strategy

t_costs_pd_d_comp Total discounted costs accrued in PD health state, comparator strategy

t_costs_pd_d_int Total discounted costs accrued in PD health state, intervention strategy

t_qaly_ae_int Quality-adjusted life year decrement associated with the occurence of adverse events, intervention strategy

t_costs_ae_int Costs associated with the occurence of adverse events, intervention strategy

inc_ly Incremental QALYs obtained with the intervention versus the comparator

inc_qaly Incremental QALYs obtained with the intervention versus the comparator

inc_costs Incremental costs obtained with the intervention versus the comparator

Source

Pouwels XGLV, Kroeze K, van der Linden N, Kip MMA, Koffijberg H. Validating Health Economic Models With the Probabilistic Analysis Check dashBOARD. Value Health. 2024 Aug;27(8):1073-1084. doi: 10.1016/j.jval.2024.04.008.

Link to the original data ("df_pa_psm") used in the PACBOARD publication: https://github.com/Xa4P/pacheck/tree/master/data-raw

do_check *Perform a check*

Description

Checks whether variables fulfill a specific test.

Usage

```
do_check(
  df,
  v_vars,
  check,
  label_check,
  template_ok = "all variables are {label_check}",
  template_fail = "{var} is not {label_check}"
)
```

Arguments

df	a dataframe.
v_vars	character vector of variables on which to apply the test.
check	a function to apply to the 'vars'.
label_check	character string. Text describing the test to pass.
template_ok	character string. Text to display when a test is passed by a variable.
template_fail	character string. Text to display when a test is not passed by a variable.

Value

List containing the results of the check (checks), and a tibble of status and message for each test (messages). The list of messages in the result contains a single line if the test passed, or if a test failed for one or more variables, a line for each failure.

Examples

```
data(df_pa)
do_check(df = df_pa,
         v_vars = c("u_pfs", "u_pd"),
         check = ~ .x >= 0,
         label_check = "positive"
         )
```

do_discount_check *Perform discounted and undiscounted results check*

Description

This function performs multiple checks on user-defined columns.

Usage

```
do_discount_check(df, v_outcomes = NULL, v_outcomes_d = NULL)
```

Arguments

df	a dataframe.
v_outcomes	(a vector of) character. Name of the variables containing undiscounted outcomes of the model.
v_outcomes_d	(a vector of) character. Name of the variables containing discounted outcomes of the model.

Details

The variables contained in ‘v_outcomes‘ and ‘v_outcomes_d‘ should be in the same order.

Value

A matrix.

Examples

```
# Checking whether discounted QALYs are lower than undiscounted QALYs using the example data
do_discount_check(df = df_pa,
                  v_outcomes = "t_qaly_comp",
                  v_outcomes_d = "t_qaly_d_comp")
```

do_quick_check *Perform quick checks of inputs and outputs*

Description

This function performs multiple checks on user-defined columns.

Usage

```
do_quick_check(
  df,
  v_probs = NULL,
  v_utilities = NULL,
  v_costs = NULL,
  v_hr = NULL,
  v_rr = NULL,
  v_r = NULL,
  v_outcomes = NULL
)
```

Arguments

df	a dataframe.
v_probs	(a vector of) character. Name of variables containing probabilities.
v_utilities	(a vector of) character. Name of the variables containing utility values.
v_costs	(a vector of) character. Name of the variables containing cost estimates.
v_hr	(a vector of) character. Name of the variables containing hazard ratios.
v_rr	(a vector of) character. Name of the variables containing relative risks.
v_r	(a vector of) character. Name of the variables containing rates.
v_outcomes	(a vector of) character. Name of the variables containing outcomes of the model.

Value

A matrix.

Examples

```
# Checking costs and utility values of the example data

do_quick_check(df = df_pa,
               v_utilities = c("u_pfs", "u_pd"),
               v_costs = c("c_pfs", "c_pd")
               )
```

---

estimate_decision_sensitivity

Estimate decision sensitivy DSA using linear metamodel

Description

This function performs a logistic regression analysis and determines the decision sensitivity to parameter value using the logistic regression. (STILL IN DEVELOPMENT)

Usage

```
estimate_decision_sensitivity(df, y, x, y_binomial = FALSE, limit = 0)
```

Arguments

df	a dataframe. This dataframe should contain both dependent and independent variables.
y	character. Name of the output variable in the dataframe. This will be the dependent variable of the logistic regression model.
x	character or a vector for characters. Name of the input variable in the dataframe. This(these) will be the independent variable(s) of the logistic regression model.
y_binomial	logical. Is 'y' already a binomial outcome? Default is 'FALSE.' If 'TRUE', the 'y' variable will be used as such, otherwise, the 'y' variable will be converted to a binomial variable using the 'limit' argument.
limit	numeric. Determines the limit when outcomes from 'y' are categorised as 'success' (1) or not (0).

Details

The method for these analyses is described in [Merz et al. 1992](https://doi.org/10.1177

Value

A dataframe with the parameter values of the fitted logistic regression and the decision sensitivity associated with each parameter included in the logistic regression model.

Examples

```
# Determining decision sensitivity using a non-binomial outcome
data(df_pa)
df_pa$inmb <- df_pa$inc_qaly * 100000 - df_pa$inc_costs
estimate_decision_sensitivity(df = df_pa,
                              y = "inmb",
                              x = c("p_pfsd", "p_pdd"),
                              y_binomial = FALSE
                              )
```

---

fit_dist *Fit distribution to parameter*

Description

This function fits statistical distributions to a user-defined parameter.

Usage

```
fit_dist(df, param, dist = c("norm", "beta", "gamma", "lnorm"))
```

Arguments

df	a dataframe.
param	character. Name of variable of the dataframe on which to fit the distributions.
dist	character or vector of character. Determine which distribution to fit on the density plot.

Details

The available distributions are: "norm" (normal), "beta", "gamma", "lnorm" (lognormal). The arguments of the lists are "AIC" which contains the Akaike Information Criteria for each fitted distribution and "Dist_parameters" which contains the parameters of the fitted distributions. The distributions are fitted using the `fitdistrplus::fitdist()`

Value

A list with two objects:

- Statistical_fit: a dataframe containing the statistical fit criteria of the fitted distributions.
- Dist_parameters: a dataframe containing the parameter value of the fitted distributions.

Examples

```
# Fitting normal and beta distribution to the "u_pfs" variable of the example dataframe.
data(df_pa)
fit_dist(df = df_pa,
         param = "u_pfs",
         dist = c("norm", "beta"))
```

`fit_lasso_metamodel` *Fit LASSO metamodel*

Description

This function fits a lasso metamodel using the `glmnet` package.

Usage

```
fit_lasso_metamodel(
  df,
  y_var = NULL,
  x_vars = NULL,
  seed_num = 1,
  standardise = FALSE,
  tune_plot = TRUE,
  x_poly_2 = NULL,
  x_poly_3 = NULL,
  x_exp = NULL,
```

```
  x_log = NULL,
  x_inter = NULL
)
```

Arguments

df	a dataframe.
y_var	character. Name of the output variable in the dataframe. This will be the dependent variable of the metamodel.
x_vars	character or a vector for characters. Name of the input variable(s) in the dataframe. This will be the independent variable of the metamodel.
seed_num	numeric. Determine which seed number to use to split the dataframe in fitting an validation sets.
standardise	logical. Determine whether the parameter of the linear regression should be standardised. Default is FALSE.
tune_plot	logical. Determine whether the plot of the results of tuning the lambda should be shown.
x_poly_2	character. character or a vector for characters. Name of the input variable in the dataframe. These variables will be exponentiated by factor 2.
x_poly_3	character. character or a vector for characters. Name of the input variable in the dataframe. These variables will be exponentiated by factor 3.
x_exp	character. character or a vector for characters. Name of the input variable in the dataframe. The exponential of these variables will be included in the metamodel.
x_log	character. character or a vector for characters. Name of the input variable in the dataframe. The logarithm of these variables will be included in the metamodel.
x_inter	character. character or a vector for characters. Name of the input variables in the dataframe. This vector contains the variables for which the interaction should be considered. The interaction terms of two consecutive variables will be considered in the linear model; hence, the length of this vector should be even.

Value

A list contaning the following elements:

- An object of the fitted metamodel,
- The coefficient of the fitted metamodel,
- information on the data used to fit the metamodel and its form.

Examples

```
#Fit lasso metamodel with two variables using the probabilistic data
data(df_pa)
fit_lasso_metamodel(df = df_pa,
                y_var = "inc_qaly",
                x_vars = c("p_pfsd", "p_pdd"),
```

```
                tune_plot = TRUE
                )
```

`fit_lm_metamodel` *Fit linear metamodel*

Description

This function fits and provides summary statistics of a linear regression model fitted on the input and output values of a probabilistic analysis.

Usage

```
fit_lm_metamodel(
  df,
  y_var = NULL,
  x_vars = NULL,
  standardise = FALSE,
  partition = 1,
  seed_num = 1,
  validation = FALSE,
  folds = 5,
  show_intercept = FALSE,
  x_poly_2 = NULL,
  x_poly_3 = NULL,
  x_exp = NULL,
  x_log = NULL,
  x_inter = NULL
)
```

Arguments

`df`	a dataframe.
`y_var`	character. Name of the output variable in the dataframe. This will be the dependent variable of the metamodel.
`x_vars`	character or a vector for characters. Name of the input variable in the dataframe. This will be the independent variable of the metamodel.
`standardise`	logical. Determine whether the parameter of the linear regression should be standardised. Default is FALSE.
`partition`	numeric. Value between 0 and 1 to determine the proportion of the observations to use to fit the metamodel. Default is 1 (fitting the metamodel using all observations).
`seed_num`	numeric. Determine which seed number to use to split the dataframe in fitting and validation sets.

validation	logical or character. Determine whether to validate the linear model. Choices are "test_train_split" and "cross_validation".
folds	numeric. Number of folds for the cross-validation. Default is 5.
show_intercept	logical. Determine whether to show the intercept of the perfect prediction line (x = 0, y = 0). Default is FALSE.
x_poly_2	character. character or a vector for characters. Name of the input variable in the dataframe. These variables will be exponentiated by factor 2.
x_poly_3	character. character or a vector for characters. Name of the input variable in the dataframe. These variables will be exponentiated by factor 3.
x_exp	character. character or a vector for characters. Name of the input variable in the dataframe. The exponential of these variables will be included in the metamodel.
x_log	character. character or a vector for characters. Name of the input variable in the dataframe. The logarithm of these variables will be included in the metamodel.
x_inter	character. character or a vector for characters. Name of the input variables in the dataframe. This vector contains the variables for which the interaction should be considered. The interaction terms of two consecutive variables will be considered in the linear model; hence, the length of this vector should be even.

Details

Standardisation of the parameters is obtained by

$$(x - u(x))/sd(x)$$

where x is the variable value, $u(x)$ the mean over the variable and $sd(x)$ the standard deviation of x.

For more details, see Jalal H, Dowd B, Sainfort F, Kuntz KM. Linear Regression Metamodeling as a Tool to Summarize and Present Simulation Model Results. Medical Decision Making. 2013;33(7):880-890. doi:10.1177/0272989X13492014

Value

A list containing the fit of the model and validation estimates and plots when selected.

Examples

```
# Fitting linear meta model with two variables using the probabilistic data
data(df_pa)
fit_lm_metamodel(df = df_pa,
                 y_var = "inc_qaly",
                 x_vars = c("p_pfsd", "p_pdd")
                 )
```

`fit_rf_metamodel` *Fit random forest metamodel*

Description

This function fits a random forest metamodel using the `randomForestSRC` package.

Usage

```
fit_rf_metamodel(
  df,
  y_var = NULL,
  x_vars = NULL,
  ntree = 500,
  seed_num = 1,
  tune = FALSE,
  var_importance = FALSE,
  pm_plot = FALSE,
  pm_vars = x_vars[1],
  validation = FALSE,
  folds = 5,
  show_intercept = FALSE,
  partition = 1,
  fit_complete_model = TRUE
)
```

Arguments

`df`	a dataframe.
`y_var`	character. Name of the output variable in the dataframe. This will be the dependent variable of the metamodel.
`x_vars`	character or a vector for characters. Name of the input variable(s) in the dataframe. This will be the independent variable of the metamodel.
`ntree`	Number of trees to grow.
`seed_num`	numeric. Determine which seed number to use to split the dataframe in fitting an validation sets.
`tune`	logical. Determine whether nodesize and mtry should be tuned. Nodesize is the minimum size of terminal nodes, mtry is number of variables to possibly split at each node. If FALSE, nodesize = 15 (for regression), and mtry = number of x-variables / 3 (for regression). Default is FALSE.
`var_importance`	logical or character. Determine whether to compute variable importance (TRUE/FALSE), or how to compute variable importance (permute/random/anti). Default is FALSE. TRUE corresponds to "anti".
`pm_plot`	logical or character. Determine whether to plot the partial ("partial") or marginal ("marginal") effect or both ("both") of an x-variable (which is denoted by pm_vars). Default is FALSE. TRUE corresponds to "both".

pm_vars	character. Name of the input variable(s) for the partial/marginal plot. Default is the first variable from the x_vars.
validation	logical or character. Determine whether to validate the RF model. Choices are "test_train_split" and "cross-validation". TRUE corresponds to "cross-validation", default is FALSE.
folds	numeric. Number of folds for the cross-validation. Default is 5.
show_intercept	logical. Determine whether to show the intercept of the perfect prediction line (x = 0, y = 0). Default is FALSE.
partition	numeric. Value between 0 and 1 to determine the proportion of the observations to use to fit the metamodel. Default is 1 (fitting the metamodel using all observations).
fit_complete_model	logical. Determine whether to fit the (final) full model. So the model trained on all available data (as opposed to the model used in validation which is trained on the test data).

Value

A list containing the following elements:

- fit: a list, see `randomForestSRC::rfsrc()` for a description of the outputs contained in this list.
- model_info: a list containing the following elements:
  - x_vars: vector of names of parameters included in the metamodel;
  - y_var: name outcome variable;
  - form: formula of the metamodel based on ‘x_vars‘ and ‘y_var‘;
  - data: dataframe containing the inputs and output values used to fit (and fit) the metamodel;
  - type: "rf" for "random forest".
- (if ‘tune‘ = TRUE) tune_fit: a list containing the results of the tuning process, see `randomForestSRC::tune()` for a description of the elements containd in this list.
- (if ‘tune‘ = TRUE) tune_plot: plot showing the out-of-bag error for each tested combination of ’mtry’ and ’nodesize’.
- (if validation != FALSE) stats_validation: data frame containing the R-squared, Mean absolute error, Mean relative error, Mean squared error in the test validation set.
- (if validation = "test_train_split") calibration_plot: plot showing the rf-predicted versus observed output values in the test validation set.

If ‘var_importance‘ is set to TRUE, the variable importance plot is printed in the console. If ‘pm_plot‘is used, the marginal/ partial importance plot(s) - drawn using `randomForestSRC::plot.variable.rfsrc()` - is (are) printed in the console.

Examples

```
# Fitting and tuning a random forest meta model with two variables using the example data
data(df_pa)
```

```
fit_rf_metamodel(df = df_pa,
                 y_var = "inc_qaly",
                 x_vars = c("p_pfsd", "p_pdd"),
                 tune = FALSE
                 )
```

`generate_cor` *Generate correlation matrix*

Description

This function generates the correlation matrix of input and output values of a probabilistic analysis.

Usage

```
generate_cor(df, vars = NULL, figure = FALSE, digits = 3)
```

Arguments

`df`	a dataframe. This dataframe contains the probabilistic inputs and outputs of the health economic model.
`vars`	a vector of strings. Contains the name of the variables to include in the correlation matrix. Default is NULL meaning all variables will be included.
`figure`	logical. Should the correlation matrix be plotted in a figure? Default is FALSE (no figure generated).
`digits`	integer. Number of decimals to display in correlation matrix. Default is 3.

Value

If figure == FALSE: a matrix with summary statistics for the selected inputs and outputs. If figure == TRUE: a tile ggplot2 of the correlation matrix.

Examples

```
# Generating summary data of all inputs using the example dataframe
data(df_pa)
generate_cor(df_pa)
```

generate_det_inputs *Generate deterministic model inputs.*

Description

This function generates the deterministic model inputs for the example health economic model developed to test the functionalities of the package.

Usage

```
generate_det_inputs()
```

Value

A list. A description of the inputs parameters is available in the documentation of the df_pa dataframe.

Examples

```
# Generating deterministic model inputs and storing them in an object.
l_inputs_det <- generate_det_inputs()
```

generate_pa_inputs *Generate probabilistic model inputs.*

Description

This function generates the probabilistic model inputs for the example health economic model developed to test the functionalities of the package.

Usage

```
generate_pa_inputs(n_sim = 10000, sd_var = 0.2, seed_num = 452)
```

Arguments

n_sim	integer. Number of probabilistic value to draw for each model input. Default is 10,000.
sd_var	numeric. Determines the standard error of the mean to use for the normal distributions when the standard error not known. Default is 0.2 (20%).
seed_num	integer. The seed number to use when drawing the probabilistic values. Default is 452.

Value

A dataframe. A description of the variables of the returned dataframe is available in the documentation of the df_pa dataframe.

Examples

```
# Generating deterministic model inputs and storing them in an object.
df_inputs_prob <- generate_pa_inputs()
```

`generate_pa_inputs_psm` *Generate probabilistic model inputs for partitioned survival model.*

Description

This function generates the probabilistic model inputs for the example health economic model developed to test the functionalities of the package.

Usage

```
generate_pa_inputs_psm(n_sim = 10000, sd_var = 0.2, seed_num = 452)
```

Arguments

`n_sim`	integer. Number of probabilistic value to draw for each model input. Default is 10,000.
`sd_var`	numeric. Determines the standard error of the mean to use for the normal distributions when the standard error not known. Default is 0.2 (20%).
`seed_num`	integer. The seed number to use when drawing the probabilistic values. Default is 452.

Value

A dataframe. A description of the variables of the returned dataframe is available in the documentation of the `df_pa_psm` dataframe.

Examples

```
# Generating probabilistic model inputs and storing them in an object.

df_inputs_prob <- generate_pa_inputs_psm(n_sim = 10)
```

`generate_sum_stats` *Generate summary statistics*

Description

This function generates summary statistics of input and output values of a probabilistic analysis.

Usage

```
generate_sum_stats(df, vars = NULL)
```

Arguments

`df`	a dataframe. This dataframe contains the probabilistic inputs and outputs of the health economic model.
`vars`	a vector of strings. Contains the name of the variables to include in the summary statistics table. Default is NULL meaning all variables will be included.

Value

A dataframe with summary statistics for the selected variables. The returned summary statistics are:

- Mean
- Standard deviation
- 2.5th percentile
- 97.5th percentile
- Minimum
- Maximum
- Median
- Skewness
- Kurtosis

Examples

```
# Generating summary data of all inputs
data(df_pa)
df_summary <- generate_sum_stats(df_pa)
```

l_psa_aaa *A dataframe containing probabilistic inputs for testing*

Description

A list containing 1,000 sets of probabilistic inputs and outputs from a discrete event simulation developed in R.

Usage

```
l_psa_aaa
```

Format

The dataframe contains the 1,000 probabilistic inputs and outputs obtained by executing the Main file_men 30years_FullModel.R' script (for 1,000 individuals) under the 'models' folder on the github repository of the model.

Source

Sweeting MJ, Masconi KL, Jones E, Ulug P, Glover MJ, Michaels JA, Bown MJ, Powell JT, Thompson SG. Analysis of clinical benefit, harms, and cost-effectiveness of screening women for abdominal aortic aneurysm. Lancet. 2018 Aug 11;392(10146):487-495. doi: 10.1016/S0140-6736(18)31222-4.

Link to Github repository: https://github.com/mikesweeting/AAA_DES_model

perform_dowsa *Perform deterministic one-way sensitivity analyses using probabilistic inputs and outputs.*

Description

This function performs the deterministic one-way sensitivity analyses (DOWSA) using probabilistic inputs and outputs for the health economic model developed to test the package. The outcome of the DOWSA is the incremental net monetary benefit.

Usage

```
perform_dowsa(df, vars, wtp = 120000)
```

Arguments

df	a dataframe. This dataframe contains the probabilistic inputs and outputs of the health economic model.
vars	a vector of strings. Contains the name of the variables for which to perform the deterministic one-way sensitivity analysis.
wtp	numeric. The willingness to pay per QALY in euros. Default is 120,000 euros per QALY.

Value

A dataframe. The outcome of the deterministic one-way sensitivity analyses is the iNMB by default.

Examples

```
# Perform the deterministic one-way sensitivity analyses for a selection of parameters

data(df_pa)
df_res_dowsa <- perform_dowsa(df = df_pa,
                              vars = c("rr", "c_pfs"))
```

---

perform_simulation *Perform the health economic simulation.*

Description

This function performs the simulation of the health economic model developed to test the functionalities of the package.

Usage

```
perform_simulation(l_params)
```

Arguments

l_params	list. List of inputs of the health economic model

Value

A vector. This vector contains the (un)discounted intermediate and final outcomes of the health economic model.

Examples

```
# Perform the simulation using the deterministic model inputs
l_inputs_det <- generate_det_inputs()
v_results_det <- perform_simulation(l_inputs_det)
```

perform_simulation_psm *Perform the health economic simulation using partitioned survival model.*

Description

This function performs the simulation of the partitioned survival health economic model developed to test the functionalities of the package.

Usage

```
perform_simulation_psm(l_params, min_fct = TRUE)
```

Arguments

l_params	list. List of inputs of the health economic model.
min_fct	logical. Should a minimum function be used to ensure PFS remains lower than OS? Default is TRUE.

Value

A vector. This vector contains the (un)discounted intermediate and final outcomes of the health economic model.

Examples

```
# Perform the simulation using one iteration of the probabilistic model inputs
l_inputs_det <- as.list(generate_pa_inputs_psm(n_sim = 1))
v_results_det <- perform_simulation_psm(l_inputs_det)
```

plot_ce *Plotting cost-effectiveness plane for two strategies.*

Description

This function plots the cost-effectiveness plane for two strategies.

Usage

```
plot_ce(df, e_int, e_comp, c_int, c_comp, currency = "euro", axes = TRUE)
```

Arguments

df	a dataframe.
e_int	character. Name of variable of the dataframe containing total effects of the intervention strategy.
e_comp	character. Name of variable of the dataframe containing total effects of the comparator strategy.
c_int	character. Name of variable of the dataframe containing total costs of the intervention strategy.
c_comp	character. Name of variable of the dataframe containing total costs of the comparator strategy.
currency	character. Default is "euro". Determines the currency sign to use in the incremental cost effectiveness plane. Currently included signs: "euro", "dollar", "yen", "none".
axes	logical. Default is TRUE, axes are plotted at x = 0 and y = 0. If FALSE, no axes are plotted.

Value

A ggplot2 graph.

Examples

```
# Plot cost effectiveness plane
data("df_pa")
plot_ce(df = df_pa,
        e_int = "t_qaly_d_int",
        e_comp = "t_qaly_d_comp",
        c_int = "t_costs_d_int",
        c_comp = "t_costs_d_comp",
        currency = "none"
        )
```

---

plot_ce_mult *Plotting cost-effectiveness plane.*

Description

This function plots the cost-effectiveness plane for an infinite amount of strategies .

Usage

```
plot_ce_mult(df, outcomes, costs, ellipse = FALSE, currency = "euro")
```

Arguments

df	a dataframe.
outcomes	character. Vector of variable names containing the outcomes to be plotted on the x-axis. The variable names should be structured as follows: 't_qaly_d_' followed by the name of the strategy: e.g. 't_qaly_d_intervention'.
costs	character. Vector of variable names containing the costs to be plotted on the y-axis. The variable names should be structured as follows: 't_costs_d_' followed by the name of the strategy: e.g. 't_costs_d_intervention'.
ellipse	logical. Determines whether plot should plot the dots of each iteration (default, ellipse = FALSE), or whether the mean outcomes and costs and their 95procent confidence ellipses should be plotted (TRUE).
currency	character. Default is "euro". Determines the currency sign to use in the incremental cost effectiveness plane. Currently included signs: "euro", "dollar", "yen", "none".

Value

A ggplot2 graph. # Plot cost effectiveness plane as ellipses data("df_pa") df_pa$t_qaly_d_int2 <- df_pa$t_qaly_d_int * 1.5 # creating additional outcome variable df_pa$t_costs_d_int2 <- df_pa$t_costs_d_int * 1.5 # creating additional cost variable plot_ce_mult(df = df_pa, outcomes = c("t_qaly_d_int", "t_qaly_d_comp", "t_qaly_d_int2"), costs = c("t_costs_d_int","t_costs_d_comp", "t_costs_d_int2"), ellipse = TRUE, currency = "none")

---

plot_ceac *Plotting the cost-effectiveness acceptability curves.*

Description

This function plots cost-effectiveness acceptability curves.

Usage

```
plot_ceac(df, name_wtp, currency = "euro")
```

Arguments

df	a dataframe obtained through the 'calculate_ceac()' or 'calculate_ceac_mult()'.
name_wtp	character. Name of variable of the dataframe containing the willingness-to-pay thresholds at which the probability of cost effectiveness have been defined.
currency	character. Default is "euro". Determines the currency sign to use in the incremental cost effectiveness plane. Currently included signs: "euro", "dollar", "yen", "none".

Value

A ggplot2 graph.

Examples

```
# Plot CEAC based on results from calculate_ceac()
data("df_pa")
df_ceac_p <- calculate_ceac(df = df_pa,
                            e_int = "t_qaly_d_int",
                            e_comp = "t_qaly_d_comp",
                            c_int = "t_costs_d_int",
                            c_comp = "t_costs_d_comp")
plot_ceac(df = df_ceac_p,
          name_wtp = "WTP_threshold",
          currency = "none")
```

`plot_convergence` *Plot moving average*

Description

This function plots the moving average of a user-defined variable.

Usage

```
plot_convergence(
  df,
  param,
  block_size = 500,
  conv_limit = 0,
  y_min = NULL,
  y_max = NULL,
  breaks = NULL,
  variance = FALSE
)
```

Arguments

`df`	a dataframe.
`param`	character string. Name of variable of the dataframe for which to plot the moving average.
`block_size`	numeric. Define the size of the blocks at which the mean of the variable (‘param‘) has to be defined and plotted. Default is 500 iterations.
`conv_limit`	numeric. Define the convergence limit, under which the relative change between block of iterations should lie.
`y_min`	numeric. Define the minimum value of the parameter to display on th y-axis of the convergence plot.If NULL (default, not defined), this will automatically be set near the minimum value of ‘param‘.
`y_max`	numeric. Define the maximum value of the parameter to display on th y-axis of the convergence plot. If NULL (default, not defined), this will automatically be set near the maximum value of ‘param‘.

breaks	numeric. Number of iterations at which the breaks should be placed on the plot. Default is NULL, hence a tenth of the length of the vector ‘param‘ is used.
variance	logical. Determine whether the variance of the vector should be plotted instead of the mean. Default is FALSE.

Value

A ggplot graph.

Examples

```
# Checking the moving average of the incremental QALYs using the example data.
data(df_pa)
plot_convergence(df = df_pa,
                 param = "inc_qaly"
                 )
```

---

plot_ice *Plotting the incremental cost-effectiveness plane.*

Description

This function plots the incremental cost-effectiveness plane for two strategies.

Usage

```
plot_ice(
  df,
  e_int,
  e_comp,
  c_int,
  c_comp,
  col = NULL,
  n_it = NULL,
  wtp = NULL,
  currency = "euro",
  axes = TRUE
)
```

Arguments

df	a dataframe.
e_int	character. Name of variable of the dataframe containing total effects of the intervention strategy.
e_comp	character. Name of variable of the dataframe containing total effects of the comparator strategy.

c_int	character. Name of variable of the dataframe containing total costs of the intervention strategy.
c_comp	character. Name of variable of the dataframe containing total costs of the comparator strategy.
col	character. Name of variable of the dataframe to use to colour (in blue) the plotted dots. Default is NULL which results in grey dots.
n_it	(vector of) numeric value(s). Designate which iteration should be coloured in the colour red.
wtp	numeric. Default is NULL. If different than NULL, plots a linear line with intercept 0 and the defined slope.
currency	character. Default is "euro". Determines the currency sign to use in the incremental cost effectiveness plane. Currently included signs: "euro", "dollar", "yen", "none".
axes	logical. Default is TRUE, axes are plotted at x = 0 and y = 0. If FALSE, no axes are plotted.

Value

A ggplot2 graph.

Examples

```
# Generating plot using the example dataframe, and a willlingness-to-pay threshold of 80,0000 euros.
data(df_pa)
plot_ice(df = df_pa,
         e_int = "t_qaly_d_int",
         e_comp = "t_qaly_d_comp",
         c_int = "t_costs_d_int",
         c_comp = "t_costs_d_comp",
         wtp = 8000,
         currency = "none")
```

---

plot_nb *Plot (i)NMB or (i)NHB.*

Description

This function plots the Net Monetary Benefits (NMB) and Net Health Benefits (NHB) for each strategy and the incremental NMB and NHB (only pairwise comparison).

Usage

```
plot_nb(df, NMB = TRUE, comparators = TRUE, incremental = FALSE)
```

Arguments

df	a dataframe obtained through ‘calculate_nb()‘
NMB	logical. Should the (i)NMBs be plotted? Default is TRUE, if FALSE, (i)NHBs are plotted.
comparators	logical. Should the NMB/NHB for each comparator be plotted? Default is TRUE.
incremental	logical. Should the incremental NMB/NHB be plotted? Default is FALSE

Details

The use this function, the dataframe ‘df‘ should contain the variables ‘NMB_int‘, ‘NMB_comp‘, ‘iNMB‘, ‘NHB_int‘, ‘NHB_comp‘, and ‘iNHB‘. For instance, use the calculate_nb function to calculate these outcomes.

Value

A ggplot2 graph.

Examples

```
# Calculate NB's at a willingness-to-pay threshold of 80000 per unit of effects
data("df_pa")
df_nmb <- calculate_nb(df = df_pa,
             e_int = "t_qaly_d_int",
             e_comp = "t_qaly_d_comp",
             c_int = "t_costs_d_int",
             c_comp = "t_costs_d_comp",
             wtp = 80000)

# Plot NMB's for each comparator
plot_nb(df = df_nmb,
        NMB = TRUE,
        comparators = TRUE)
```

---

plot_nb_mult *Plot NMB or NHB.*

Description

This function plots the Net Monetary Benefits (NMB) and Net Health Benefits (NHB) for an infinite amount of strategies.

Usage

```
plot_nb_mult(df, outcomes, costs, wtp, NMB = TRUE)
```

Arguments

df	a dataframe.
outcomes	character. Vector of variable names containing the outcomes to be plotted on the x-axis. The variable names should be structured as follows: 't_qaly_d_' followed by the name of the strategy: e.g. 't_qaly_d_intervention'.
costs	character. Vector of variable names containing the costs to be plotted on the y-axis. The variable names should be structured as follows: 't_costs_d_' followed by the name of the strategy: e.g. 't_costs_d_intervention'.
wtp	numeric. Willingness-to-pay thresholds to use for NMB and NHB calculations.
NMB	logical. Should the NMBs be plotted? Default is TRUE, if FALSE, NHBs are plotted.

Value

A ggplot2 graph.

Examples

```
# Plot NMB's at a 50,0000 euro WTP threshold for three strategies
data("df_pa")
df_pa$t_qaly_d_int2 <- df_pa$t_qaly_d_int * 1.5
df_pa$t_costs_d_int2 <- df_pa$t_costs_d_int * 1.5
plot_nb_mult(df = df_pa,
             outcomes = c("t_qaly_d_int2", "t_qaly_d_int", "t_qaly_d_comp"),
             costs = c("t_costs_d_int", "t_costs_d_int2", "t_costs_d_comp"),
             wtp = 50000)
```

---

plot_surv_mod *Plot parametric survival models*

Description

This function plots two parametric survival models based on he functional form of the model and their parameters.

Usage

```
plot_surv_mod(
  df,
  surv_mod_1,
  surv_mod_2,
  v_names_param_mod_1,
  v_names_param_mod_2,
  label_surv_1 = "first survival",
  label_surv_2 = "second survival",
  iteration,
  time = seq(0, 5, 1)
)
```

Arguments

df	a dataframe.
surv_mod_1	character. Name of the parametric model to use for the first survival model.
surv_mod_2	character. Name of the parametric model to use for the second survival model.
v_names_param_mod_1	(vector of) character. Name of the columns containing the parameter values for the first survival model.
v_names_param_mod_2	(vector of) character. Name of the columns containing the parameter values for the second survival model.
label_surv_1	character vector. The label to provide to the first survival curve (relevant for export).
label_surv_2	character vector. The label to provide to the second survival curve (relevant for export).
iteration	integer. The row number of the iterations for which the parametric survival models have to be plotted.
time	a numerical vector. Determine at which time points survival probabilities have to be estimated for both survival models. For each of these time points, it will be checked whether the first survival model results in higher survival probabilities than the second survival model.

Details

The parametric models that can be used are the following: exponential (exp), Weibull (weibull), gamma (gamma), loglogistic (logis), and lognormal (lnorm). All these functions are implemented following their distribution function as documented in the stats package.

Value

A ggplot object.

---

predict_metamodel *Predict using a fitted metamodel*

Description

Predict using a fitted metamodel

Usage

```
predict_metamodel(model = NULL, inputs = NULL, output_type = "vector")
```

Arguments

model	model object. Built using a function from the PACHECK package.
inputs	dataframe or vector. When choosing a vector in the case of a three-variable model: the first, second, third, and fourth value represent the input for the first, second, third, and FIRST variable, respectively. Default gives the predictions based on the training data.
output_type	character. Choose an output: 'dataframe', 'long_df' (long data.frame) or 'vector'.

Value

returns a vector of the the predictions ('vector' output_type) or the parameter values used for the predictions and the predictions ('dataframe' or 'long_df' output_type).

Examples

```
#Making 3 predictions for a two-variable metamodel,
# using a vector as input, and yielding a dataframe as output.
data(df_pa)
lm_fit = fit_lm_metamodel(df = df_pa,
                 y_var = "inc_qaly",
                 x_vars = c("p_pfsd", "p_pdd")
                 )

vec = c(0.1,0.2,0.08,0.15,0.06,0.25)

predict_metamodel(model = lm_fit,
                 inputs = vec,
                 output_type = "dataframe"
                 )
```

---

summary_ice *Summary statistics of the incremental cost-effectiveness plane.*

Description

This function computes the probability that the probabilistic outcome is in each of the quadrants.

Usage

```
summary_ice(df, e_int, e_comp, c_int, c_comp)
```

Arguments

df	a dataframe.
e_int	character. Name of variable of the dataframe containing total effects of the intervention strategy.

e_comp	character. Name of variable of the dataframe containing total effects of the comparator strategy.
c_int	character. Name of variable of the dataframe containing total costs of the intervention strategy.
c_comp	character. Name of variable of the dataframe containing total costs of the comparator strategy.

Value

A dataframe.

Examples

```
# Generating statistics of the incremental cost-effectiveness plane using the example data.
data(df_pa)
summary_ice(df = df_pa,
            e_int = "t_qaly_d_int",
            e_comp = "t_qaly_d_comp",
            c_int = "t_costs_d_int",
            c_comp = "t_costs_d_comp"
            )
```

---

validate_metamodel *Validate metamodels*

Description

Validate metamodels

Usage

```
validate_metamodel(
  model = NULL,
  method = NULL,
  partition = 1,
  folds = 1,
  show_intercept = FALSE,
  seed_num = 1,
  df_validate = NULL
)
```

Arguments

model	model object. Built using a function from the PACHECK package.
method	character, validation method. Choices are: cross-validation ('cross_validation'), train-test split ('train_test_split'), or the user can input a new dataframe which will be used as the test-set ('new_test_set'). No default.

partition	numeric. Value between 0 and 1 to determine the proportion of the observations to use to fit the metamodel. Default is 1 (fitting the metamodel using all observations).
folds	numeric. Number of folds for the cross-validation. Default is 1 (so an error occurs when not specifying this argument when cross-validation is chosen).
show_intercept	logical. Determine whether to show the intercept of the perfect prediction line (x = 0, y = 0). Default is FALSE.
seed_num	numeric. Determine which seed number to use to split the dataframe in fitting and validation sets.
df_validate	dataframe. The dataframe to be used for validating the model. By default the dataframe used when building the model is used.

Value

.....

Examples

```
#Validating meta model with two variables using the probabilistic data, using cross-validation.
data(df_pa)
lm_fit = fit_lm_metamodel(df = df_pa,
                 y_var = "inc_qaly",
                 x_vars = c("p_pfsd", "p_pdd")
                 )

validate_metamodel(model = lm_fit,
                 method = "cross_validation",
                 folds = 5
                 )
```

---

vis_1_param *Visualise the distribution of a single parameter*

Description

This function plots the distribution of a single parameter.

Usage

```
vis_1_param(
  df,
  param = NULL,
  binwidth = NULL,
  type = "histogram",
  dist = NULL,
  user_dist = NULL,
  user_param_1 = NULL,
```

```
  user_param_2 = NULL,
  user_mean = NULL
)
```

Arguments

df	a dataframe.
param	character. Name of variable of the dataframe for which the distribution should be plotted.
binwidth	numeric. Determine the width of the bins to use, only applied in combination with "histogram". Default is 30 bins.
type	character. Determine which plot to return: "histogram" for a histogram, "density" for a density plot. Default is "histogram".
dist	character or vector of character. Determine which distribution to fit on the density plot.
user_dist	character string. User-defined distribution to fit. Default value is NULL.
user_param_1	character string. First parameter of the user-defined distribution to fit.
user_param_2	character string. Second parameter of the user-defined distribution to fit.
user_mean	numeric value. mean value to plot on the graph. Default is NULL

Details

The available distributions are: "norm" (normal), "beta", "gamma", "lnorm" (lognormal). TO CHECK –> ask for mean and SD/SE for the user-defined distribution???

Value

A ggplot2 graph.

Examples

```
# Generating histogram for the costs of progression-free health state, bins of 50 euros
data(df_pa)
vis_1_param(df = df_pa, param = "c_pfs", binwidth = 50)
```

---

vis_2_params *Visualise the distribution of two parameters*

Description

This function plots the distribution of two parameters in a scatterplot.

Usage

```
vis_2_params(
  df,
  param_1,
  param_2,
  slope = NULL,
  intercept = 0,
  check = NULL,
  fit = NULL
)
```

Arguments

df	a dataframe.
param_1	character. Name of variable of the dataframe to be plotted on the x-axis.
param_2	character. Name of variable of the dataframe to be plotted on the y-axis.
slope	numeric. Default is NULL. If different than 0, plots a linear line with a user-defined intercept and the defined slope.
intercept	numeric. Default is 0. Intercept of the user-defined slope.
check	character. Default is NULL. When set to "param_2 > param_1". The dots fulfilling the condition are coloured in red.
fit	character. Designate the type of smooth model to fit to the relation of ‘param_1‘ (x) and ‘param_2‘ (y). It can take the values "lm, "glm", "gam", and "loess". A model will be fitted according to the methods described in ggplot2::geom_smooth().

Value

A ggplot graph.

Examples

```
# Generating plot for the costs of progression-free health state versus incremental costs
data(df_pa)
vis_2_params(df = df_pa, param_1 = "c_pfs", "inc_costs")
```

Index

∗ **datasets**
    df_ckd_inputs, 14
    df_ckd_results, 14
    df_iviRA_pa, 15
    df_pa, 16
    df_pa_psm, 18
    l_psa_aaa, 34

calculate_ceac, 3
calculate_ceac_mult, 4
calculate_nb, 5, *42*
calculate_nb_mult, 6
check_binary, 7
check_mean_qol, 8
check_positive, 8
check_psa_darth, 9
check_range, 10
check_sum_probs, 11
check_sum_vars, 12
check_surv_mod, 13

df_ckd_inputs, 14
df_ckd_results, 14
df_iviRA_pa, 15
df_pa, 16, *31*
df_pa_psm, 18, *32*
do_check, 20
do_discount_check, 21
do_quick_check, 21

estimate_decision_sensitivity, 22
exp, *14*, *44*

fit_dist, 23
fit_lasso_metamodel, 24
fit_lm_metamodel, 26
fit_rf_metamodel, 28
fitdistrplus::fitdist(), *24*

gamma, *14*, *44*
generate_cor, 30
generate_det_inputs, 31
generate_pa_inputs, 31
generate_pa_inputs_psm, 32
generate_sum_stats, 33
ggplot2::geom_smooth(), *49*
glmnet, *24*

l_psa_aaa, 34
lnorm, *14*, *44*
logis, *14*, *44*

perform_dowsa, 34
perform_simulation, 35
perform_simulation_psm, 36
plot_ce, 36
plot_ce_mult, 37
plot_ceac, 38
plot_convergence, 39
plot_ice, 40
plot_nb, 41
plot_nb_mult, 42
plot_surv_mod, 43
predict_metamodel, 44

randomForestSRC, *28*
randomForestSRC::plot.variable.rfsrc(), *29*
randomForestSRC::rfsrc(), *29*
randomForestSRC::tune(), *29*

stats, *14*, *44*
summary_ice, 45

validate_metamodel, 46
vis_1_param, 47
vis_2_params, 48

weibull, *14*, *44*